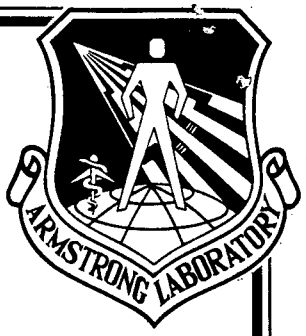
AL/HR-TR-1994-0062

DTIC
ELECTE
DEC 0 8 1994
S F D



ARMSTRONG LABORATORY

SYSTEM CONCEPT OF OPERATIONS EVALUATOR (SCOPE) FINAL TECHNICAL REPORT

Terry Raymond

MYSTECH Associates, Inc.
60 Hammarlund Way
Middletown, Rhode Island 02842-5632

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION
2698 G Street
Wright-Patterson Air Force Base, Ohio 45433-7604

19941201 115

SEPTEMBER 1994

Final Technical Report for Period February 1993 to November 1993

Approved for public release; distribution is unlimited

AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

DTIC QUALITY INSPECTED 1

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation, or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this report and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

MICHAEL J. YOUNG
Contract Monitor

BERTRAM W. CREAM, Chief
Logistics Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1994	Final - February 1993 - November 1993

4. TITLE AND SUBTITLE

System Concept of Operations Evaluator (SCOPE)
Final Technical Report

5. FUNDING NUMBERS

C - F33615-88-C-0009
PE - 62205F
PR - 1710
TA - 00
WU - 60

6. AUTHOR(S)

Terry Raymond

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MYSTECH Associates, Inc.
60 Hammarlund Way
Middletown, Rhode Island 02842-5632

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Armstrong Laboratory
Human Resources Directorate
Logistics Research Division
2698 G Street
Wright-Patterson AFB, OH 45433-7604

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL/HR-TR-1994-0062

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Michael J. Young, AL/HRGA, (513) 255-8829

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the development of the System Concept of Operator Evaluator (SCOPE). SCOPE is a modeling and simulation tool that allows analysts to evaluate concepts of operations against alternative command center designs. SCOPE is built on top of the SMALLTALK programming language as an object-oriented analysis tool. This report provides a detailed presentation of SCOPE's functionality.

14. SUBJECT TERMS		15. NUMBER OF PAGES
command and control computer science modeling	simulation	32
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

CONTENTS

Page

Preface iv
Introduction 1
System Concept of Operations Evaluator Goals & Techniques 3
Modeling Goals 3
User-Defined Model Framework 3
Object-Oriented Design 4
Visually Interactive Techniques 4
Layering 4
Usability Techniques 4
Consistency 4
Visually Interactive Methods 4
Reduced Command Set 4
Complexity-Hiding 5
Data Reuse 5
Accomplishments 6
Database Study 6
System Animation 6
Analysis and Results 6
Query Browser 7
Layer Manager 7
Improved User Interface 7
Future Issues 8
Programming 8
Paths 9
Output Formatter 9
Modify the Picture Definer 9
Reevaluate Script Builder 10
Object-Oriented Data Base Management System Future 10
General Association Objects - Records 10
Active Attributes, Attribute Triggers 11
Better Visible/Invisible Attribute Selection 11
Tiled vs. Overlapping Windows 11
User Configuration 11
Help System 11
Acronyms 12
Appendix: NORAD and USSPACECOM Command Center Study 13
References 19

Accession For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and/or Special
A-1

PREFACE

This document describes the development of the System Concept of Operations Evaluator (SCOPE) tool during Fiscal Year (FY) 1993. SCOPE was developed by Mystech Associates, Inc. of Middletown, Rhode Island. The development effort described in this document was performed, in part, for Bolt Beranek and Newman, Inc. (BBN) under subcontract No. 30585 in support of Air Force Contract No. F33615-91-D-009: Research, Development, Training and Evaluation Support (RDTE), Delivery Order #6: System Concept of Operateation Evaluator (SCOPE), funded by Armstrong Laboratory, Logistics Research Division (AL/HRG), Wright-Patterson Air Force Base, Ohio. Dr. William H. Levison of BBN served as the Technical Monitor for the subcontract to Mystech Associates, and Mr. Michael Young provided technical direction for AL/HRG. Mr. Terry Raymond of Mystech Associates was the SCOPE project engineer.

Before Armstrong Laboratory took over financial responsibility for the program in March 1991, the SCOPE technical development effort was managed by Mr. James Cary of Naval Underwater Warfare Center (NUWC) in Newport, Rhode Island. NUWC invested more than three years in the SCOPE program (then known as the Submarine Concept of Operations Evaluator) prior to Air Force involvement. That effort successfully produced an exploratory development (6.2) "proof-of-concept demonstrator" which illustrated the potential features and capabilities an operational SCOPE prototype could employ. Unfortunately, the demonstrator was "hardwired" to depict a submarine attack center and could not be used for other applications. Armstrong Laboratory's objective was to build on the concepts proven by NUWC and develop a fully functional tool that could be used in a variety of application areas outside the submarine domain.

Modeling and simulation technologies have been identified as extremely cost-effective means to answer questions about complex systems. SCOPE is an integral part of the AL/HRG efforts in this area. As a 6.2 program, SCOPE is generating technical ideas in the areas of user interface and model-building capabilities for larger 6.3 programs. As a stand-alone software tool, SCOPE has already provided a needed capability to operational users through field testing at North American Air Defense Command (NORAD) and U.S. Space Command (USSPACECOM) in Colorado Springs, Colorado.

SYSTEM CONCEPT OF OPERATIONS EVALUATOR (SCOPE) FINAL TECHNICAL REPORT

INTRODUCTION

The task of designing weapon systems and command systems is becoming increasingly difficult. The primary reason for this difficulity is the increasing sophistication of the threat environment; however, the shrinking defense budget and the addition of new missions also complicate the process. Modern-day weapon systems must be more capable, address more missions, require fewer personnel, and cost less than previous systems. To the system designer, this equates to increased complexity. Computer-Aided System Engineering (CASE) tools can assist in developing systems, but most CASE tools are designed to address only a portion of the system design. There are tools that address system requirements analysis, software design, real-time system behavior, and human performance modeling; however, these narrowly focused tools do not adequately address multiple problem domains. Furthermore, it is often difficult to transfer the modeling information from one type of model to another for reuse. Many tools do not manage data in a way that makes it easy to create alternative models. System Concept of Operations Evaluator (SCOPE) addresses these issues and emphasizes another critical feature—simplicity. Consequently, SCOPE is both powerful and easy to use.

The need to address a broad problem domain led SCOPE designers to use an object-oriented modeling paradigm to represent the information or knowledge about a system. Using this paradigm, SCOPE has become a general-purpose tool that allows users to define their own basic object types and object relationships, thereby enabling analysts to model complex systems in a manner that suits their situation. SCOPE's greatest strength is its capability to use an object-oriented representation of static models. The static model captures a considerable portion of the complete system model and serves as the foundation of the dynamic model. By itself, the static model serves as a database upon which queries and analysis can be performed. Dynamic analysis is performed by creating an object sequence for a particular scenario and running the sequence with a discrete-event simulator. The simulation results are analyzed using standard analysis techniques.

SCOPE began in Fiscal Year (FY) 87 as a Naval Underwater Systems Center (NUSC) sponsored project called Concepts Assessment for Combat Control (CACC) Encyclopedia. The purpose of the CACC Encyclopedia was to prototype a number of techniques for modeling complex systems. Initially, this was restricted to just submarine combat systems using an object-oriented representation. The project was successful in demonstrating the utility of this representation. In FY88, a new tool called the Submarine Concept of Operation Evaluation was developed. It explored various user interface and analysis concepts. In FY89, the tool was modified to investigate scenario scripting and to support the analysis of the scripts using a new knowledge representation where each binary relationship between two objects was represented as a separate object. This representation, although more flexible at representing system dynamics than the object-oriented representation, required significantly more computer resources. In FY91, the Air Force Armstrong Laboratory, Logistics Research Division (AL/HRG), funded a new effort that is the foundation of the current SCOPE tool. AL/HRG recognized the potential of the tool to represent and evaluate a much wider range of system models than a submarine combat system such as the command and control activity of a ground-based operations center.

The purpose of this effort was to build a broader, more flexible tool using the "lessons learned" from the previous versions. The broadened focus of the tool led to its new name: System Concept of Operation Evaluator.

In FY91, the Commander-in-Chief (CINC) of the two command centers at NORAD and USSPACECOM ordered a study to determine whether it would be cost-effective to combine the centers, since they had many overlapping functions and utilized common information. Also in FY91, AL/HRG began discussions with the MITRE corporation, NORAD, and USSPACECOM about using SCOPE to assist MITRE in modeling the two command centers. MITRE and command center personnel soon recognized that SCOPE was an appropriate tool to model the command centers. Consequently, AL/HRG agreed to assist MITRE in using SCOPE to create a model of the command centers, and MITRE agreed to provide user feedback to AL/HRG on the strengths and weaknesses of SCOPE. These models were to become baselines from which the impacts of crew consolidation, technology insertions, and future missions (e.g., ballistic missile defense) could be studied. However, early in FY93, the CINC directed that the two command centers be immediately consolidated, and funding for the modeling effort was withdrawn. An account of the modeling effort up to its termination is provided in the Appendix.

This report describes the goals of the SCOPE tool, the accomplishments during FY93, and issues that should be addressed in the future. It assumes the reader is familiar with the operation of the SCOPE tool. Readers who are not familiar with its operation are referred to the SCOPE User's Manual.

SCOPE GOALS & TECHNIQUES

The overarching goal of SCOPE is to provide model flexibility and ease of use. Modeling flexibility is imperative to address the needs of large complex systems analysis and evolving requirements. Ease of use is necessary to make the tool usable to personnel with differing tool experience and skill levels. The design philosophy of SCOPE is to give users only the amount of exposure to the functioning of SCOPE that is necessary to perform their tasks. Below are the techniques that permit SCOPE to achieve these goals.

Modeling Goals

The SCOPE design philosophy is embodied in three major modeling goals. These goals are presented in the following paragraphs.

- Give users a **multipurpose, generic model**. A generic tool lets users address a wider range of situations and issues than a specific tool. This brings with it, however, a correspondingly greater burden for users because they must create the model framework and analysis tools. For complex system analysis, which is characterized by a broad domain, flexibility and information reuse easily outweigh this cost.

- Focus on **information management**. This focus is critical because managing model data is the most difficult and important part of modeling complex systems.

- Provide **multilevel and multiversion modeling** capability. This capability permits ease of transition from simple to more complex models and facilitates a thorough analysis of alternative solutions.

Important modeling techniques to achieve those goals are described in the following paragraphs.

User-Defined Model Framework

A user-defined model framework consists of a data schema definition together with information about tool configuration. The configuration information comprises user-defined functions and values placed into specific object attributes. To make it easier for users to create a model framework, SCOPE is delivered with a template for a model framework of a command center. An example of how the SCOPE tool uses the template information is the maintenance of attribute values. Attributes can contain one or more values; however, users think of managing multivalue attributes differently than single-value attributes. In managing a multivalue attribute, users think of adding or removing values. But, in managing a single-value attribute, users generally think of replacing the value—not adding and removing it. Thus, to simplify the maintenance of attributes, users define whether an attribute has one or many values. Then, whenever they select a new object for an attribute, SCOPE will automatically replace the old value if it is a single-value attribute; if it is a multivalue attribute, SCOPE will add the value. The process of defining the framework is simplified by using inheritance. Users define most of the framework at the root of each hierarchy and let SCOPE fill in the subordinate definitions based upon the "parent." The framework definition process is kept separate from the data management activity. When modeling a large and complex system, a modeling expert uses the definition facility to define the framework, while the remainder of the design team uses only the simplified knowledge base management facility.

Object-Oriented Design

By its very nature, object-oriented design provides an efficiency of design that reduces the wasted time and effort entailed in the traditional model-building process. In much the same way that subroutines made conventional programming easier for the programmer, object-oriented programming permits the designer to concentrate on the product instead of the process.

Visually Interactive Techniques

A major departure from the conventional approach to data management is SCOPE's use of visually interactive techniques for data entry and modification. SCOPE does not require user input of identifying information to retrieve data sets; rather users can simply point to the desired data components and order the appropriate action by choosing commands from lists of options. Data need be entered only once by the users.

Layering

The transition from initial concept to a full-scale model is facilitated by the layering concept, in which more specific objects are overlaid upon abstract elements. For instance, specific system implementations can be overlaid upon requirements specifications. This makes it feasible to scale up to larger models without losing the design elements of the model foundation and without significant recoding of material. Furthermore, layering permits the modeler to produce models that are identical to one another except for designer-specified elements. This allows users to perform analysis on the performance of the different versions so that useful conclusions can be drawn about alternative designs.

Usability Techniques

The multi-application capability, design flexibility, and tailorability aspects of SCOPE can be made accessible only if the SCOPE tool is easy to use. If users are to make best use of the SCOPE tool, it must be intuitive in its operation, have consistency of function and appearance, and provide assistance in the areas where the operator may encounter difficulty.

Consistency

Common visual formats have been employed throughout the SCOPE tool. The concepts and operations of the tools within SCOPE have been made as consistent as is practicable. The object examiner, class hierarchy list view, and pull-down menu bar (for example) are predictably placed and operate identically when the operator changes between functions within SCOPE.

Visually Interactive Methods

The SCOPE tool provides users with visual cues—a visually interactive environment—and a help system when a process is nonintuitive or difficult to comprehend. Extensive use has been made of familiar concepts and techniques, incorporating the best features of the Windows and Macintosh environments. "Point and click" techniques, pull-down menus, lists of options and choices, and other concepts have been borrowed from those environments, greatly reducing the amount of training required and putting users to work almost immediately.

Reduced Command Set

The number of commands and options are kept to an irreducible minimum, consistent with providing breadth of application and flexibility.

Complexity-Hiding

Complexity-hiding permits users of varying experience and skill levels to make effective use of SCOPE without exposing them to a greater level of detail than needed. The knowledgeable end-user can become more involved in adjusting the operation of the model, changing its features, and making changes to the design itself. Other, less-knowledgeable users, such as domain experts, senior managers, or others with less technical expertise, may use SCOPE with minimum exposure to its overall complexity, performing only those actions associated with their immediate task.

Data Reuse

Throughout the design life cycle of a large, complex system, the design team performs many types of analysis on the system and its components. All too often in conventional design, the information used in one analysis might be used in another, but is not. This sometimes occurs because individual analysts were unaware that another analyst used identical or similar information or processes. SCOPE allows most of the analysis to be done with one modeling environment, thereby facilitating the reuse of information.

ACCOMPLISHMENTS

Database Study

Some important issues for the long-term viability of the SCOPE tool are its ability to scale up to large models, exchange information with other computer-based tools, and operate in a networked environment. To address these issues, a study was performed to review the applicability of various commercial Object-Oriented Data Base Management Systems (OODBMS) and database/data exchange standards. When the survey was completed, only two OODBMSs provided specific Smalltalk support. One of them, Gemstone, is too expensive and the other, Versant, does not appear to have the features necessary to support SCOPE, although a more detailed analysis may prove otherwise. Other vendors indicated they have plans to provide Smalltalk support sometime in 1994. The survey concluded that it is too early to recommend a OODBMS for SCOPE. The object-oriented database standards are also too immature to provide much assistance. Most data exchange standards predate the development of object-oriented databases and are partially incompatible with SCOPE's object-oriented representation. Some vendors have used the Portable Common Tools Environment (PCTE) standard to exchange object-oriented data, so it is a good candidate for a more detailed investigation.

System Animation

The capability to animate either the system diagram (a layout diagram of the physical system) or the script graph of a scripted scenario has been added. When the system diagram is animated, each component is highlighted as the script specifies that component is to perform an activity. Also, lines are drawn in the diagram between components to indicate information being delivered to the performing component. The graph animation simply highlights each activity and information node as they are activated in the script. A panel is also provided to allow users to control the animation.

Analysis and Results

Several analysis and display tools (principally spreadsheets) were surveyed. These general purpose tools provide basic analysis functions and a programming language with which users define the analysis. They also provide various types of plotting and charting for the display of analysis results. Some of the more sophisticated programs allow users to create analysis windows. The predefined analysis capability of a general purpose analysis tool is limited to performing relatively simple, domain-independent analyses such as linear regression. However, users usually require more complex, domain-specific analysis tools. Thus, it is more cost-effective to enable users to construct a panel to control the desired analysis, be it complicated or simple. This capability to create user-defined analysis panels is already provided by the Picture Definer in conjunction with the user-defined functions. However, the ability to plot the results was missing. It has now been incorporated. A plot can depict the data from one or more attributes and may have multiple axes so data can be overlaid. Additionally, the data can be plotted with different line patterns, line colors, data point symbols, and data point symbol colors.

Query Browser

The ability to perform ad hoc model queries is provided by the Query Browser, using the concept that a query is an operation upon one or more sets of objects. The browser enables users to define the contents of two sets. A query operates on one or both of these sets can then be entered. The query can be a type of data extraction or filter function that creates a new set of objects that can be used for further queries. The query is written as a Smalltalk expression—a job that is about as difficult as writing a Structured Query Language (SQL) query. Smalltalk already provides SQL-like operations (such as "select:," "reject:," and "intersection:"); therefore, adding a new SQL language was not necessary. Furthermore, using Smalltalk allows users to execute a user-defined function from within a query, providing them with great power and flexibility.

Layer Manager

The structure of a model changes as users/designers learn more about their problem domain and the types of analysis that are performed on the model. The layer manager lets users redefine each layer in a model as part of this evolutionary process. Even though it is bad practice to redefine the layers after much model development has occurred, it inevitable that some modifications and adjustments will be made. The Layer Manager permits such redefinition of the layers, providing the ability to move definitions from one layer to another, add new layers, save individual layers, and load a new layer.

The current version of SCOPE does not specifically prohibit making illegal layer definitions. For example, users can move an object's attributes into a lower layer, even though attributes of objects should not be defined in a layer beneath the object's layer. If the (illegal) configuration is then saved, a subsequent load of the knowledge base would fail. Rules to prevent such illegal actions will be incorporated later.

Improved User Interface

The process of refining the user-to-SCOPE interface is a continuous one with many small changes being made to increase consistency and simplicity of operation.

FUTURE ISSUES

Programming

Although SCOPE would be easier to use if no user programming were required, this would reduce model flexibility and versatility. Therefore, the amount of programming required has been minimized, and the difficulty of programming has been reduced rather than eliminated altogether. Some programming is required because, during the modeling process, users need to define object behavior and perform quantitative analysis. Additionally, some of SCOPE's generic operations are tailored to a specific model via user-defined functions. Thus, the challenge to SCOPE development is to maintain the flexibility of programming while providing ease of use. The following programming approaches were examined to address this tradeoff:

- use Smalltalk as a textual programming language (i.e. no change);
- use Smalltalk with an added help facility;
- use Smalltalk as a visual language;
- change to a new, *textual* programming language; and
- change to a new, *visual* programming language.

First, "using Smalltalk" was compared to "changing to a new textual language." Smalltalk is a language with a small, simple syntax and a large library. The difficulty of using Smalltalk is actually the difficulty of using the library. Defining a new, small, simple textual language for users is equivalent to using a reduced subset of Smalltalk. The level of difficulty is identical, but the increase in cost and potential loss of flexibility is great; thus, the option of changing to a textual programming language was rejected.

Next, textual and visual programming languages were compared. Programming languages permit users to describe what operation is to be performed upon which data and how that operation and data depend upon other operations and data. A major problem in programming is describing the operation and its interdependencies in the proper level of detail. Furthermore, textual and visual programming languages use markedly different techniques.

Textual languages generally rely on the programmer to understand the dependencies among the operations. To some extent, however, dependent structures can be illustrated by text formatting (indenting, nesting, etc.) or "pretty printing."

Visual programming languages show dependencies between the various operations and data by connecting them with lines. The lines between operations show either the data flow, which implies the sequence of operations, or depict control lines that dictate sequential activity. The operations are illustrated by icons and symbols, text-in-a-box, or a combination of the two. Icons and symbols can make understanding more intuitive, but only if there are not too many of them. Text in boxes, on the other hand, can be unlimited in number but require more effort to learn. Thus, icons and symbols are effective in a limited problem domain or when augmented with text-in-a-box. For an unknown or broad problem domain, text-in-a-box augmented with character symbols is a better approach. Using lines to indicate dependencies of the operations and data works better than textual programming, but textual programming can better convey what the operation and its effects are. A visual representation requires more physical space to convey information, and as a result, there is a loss of coherence when relating information that is spatially more distant. This problem can sometimes be overcome by keeping related operations

and data close to each other or composing the descriptions hierarchically. An important advantage of visual programming languages is that they are used in a programming environment. Users select the appropriate operation or construct from a menu or palette. The visual cues provided by the palette or menu aid novices or casual users in the construction of a program.

For the SCOPE tool, with its unrestricted problem domain, the most important features of visual programming are visual cues, explicit dependency diagrams, and, to a lesser extent, hierarchical descriptions. Thus, the programming facility should be improved in phases.

The first phase is the current implementation which uses textual Smalltalk. The second phase is to add a help system to the textual Smalltalk. This action is equivalent to providing visual cues. The third phase is to provide the capability to diagram Smalltalk code. A diagram would show the data dependencies in the code. The last phase is to expand a called method within a diagram, thus, integrating Smalltalk's inherent hierarchical description into a diagram.

Paths

The schema defined by the attributes, choices, and crosslinks can be viewed as a template of a network graph. For example, refering to the template knowledge base provided with SCOPE, the "Choices" attribute specification of the "Destination" attribute in class "Information" is the "Activity" class, which has an attribute "Outputs" with its "Choices" attribute specification being the class "Information." Frequently, the need arises to traverse various attributes and objects in a model. Conceptually, the definition of a series of object/attribute pairs is a path template that can be repeatedly used to traverse the knowledge base graph. A complete definition of a path template would consist of a repeating object/attribute/filter definition and a termination condition. The details of this, particularly the avoidance of cycles, need to be worked out. Once an interface exists for defining and selecting paths, it could be used for analysis, queries, and the data output formatter.

Output Formatter

The concept behind the output formatter is to enable users to output data from SCOPE to a file in a user-defined format. The initial concept was to have something similar to a combination of the format definitions in a "C" print statement and the operation of the knowledge base pictures; however, this concept proved difficult to implement. There is a problem with traversing objects and attributes to some arbitrary limit while preventing infinite loops. The basic approach of the formatter is to define format objects that contain the formatting information. Traversal is still being worked out. It might be necessary to place limitations on what can be done in order to keep the mechanism from getting too complex. If path objects are incorporated into the SCOPE tool, they could possibly be used to define how traversal should occur.

Modify the Picture Definer

The Picture Definer has changed a good deal during SCOPE development and its user interface should be reexamined. There are, for example, a number of commands on the pull-down menus that apply to only some of the objects in a picture. It might be more appropriate to place these menu commands on separate object editor panels (as was done for the objects associated with the newly provided plots). Also, it would be easier to build an analysis or control

panel if the user could choose from a variety of buttons and "widgets." The display of analysis data could be improved if more than one plot type were available to the user. Also, plot scrolling and zooming should be provided.

Reevaluate Script Builder

The Picture Definer's user interface should be evaluated, and changes should be made to bring it up to the level of other interfaces. The static script view, which originally was the only way to construct a script, should be reevaluated. The primary mechanism for script construction now is the dynamic script view, with the static script view being relegated to an alternative way of viewing the results of a simulation run. Although the static script view can be used to modify a script created with the dynamic script view, those modifications are usually incompatible with the dynamic script view and may even make the script unusable by the dynamic script view. Thus, script modifications made in the static script view must be made compatible with the dynamic script view, or the static script view must be made a display-only view.

Within the dynamic script view, the graph layout algorithm does not do a very good job, and replacement algorithms should be sought.

Object-Oriented Data Base Management System Future

Object-oriented database management is the single area where further research will yield the greatest payoff. Therefore, it is imperative that further research be performed into OODBMSs that are compatible with Smalltalk. Currently, an OODBMS—Tensengrity from Polymorphic—is in beta test for VisualWorks/Smalltalk (which is ParcPlace's Smalltalk with an integrated GUI builder). The vendor plans to release Tensengrity for the Windows, OS/2, Mac, and SUN Unix platforms in January 1994. Today, it is supplied only for Digitalk's Smalltalk/V. The release price will be about $500.00. It is recommended that this OODBMS be obtained and integrated into SCOPE. While it may not be the complete solution for SCOPE, the process of integrating an OODBMS into SCOPE would provide great insight into which OODBMS features are important to SCOPE. If this OODBMS works with SCOPE, it will be a tremendous advantage because of its portability; none of the other OODBMSs provide the same degree of cross-platform portability.

Another issue related to OODBMSs is exchanging data with other tools. The PCTE standard appears to be the most promising data exchange standard for SCOPE to use. Therefore, the PCTE standard—and also the standards from the Object Management Group (OMG) and Object Data Management Group (ODMG)—should be closely examined to determine if SCOPE could use one of them for effective data exchange.

General Association Objects - Records

Some modeling applications require that several objects be associated with one another, as is frequently done with records. Currently, SCOPE does not adequately provide this capability. If SCOPE were to provide a record-type of object, it would probably be an extension of the KBObjectInstances. It is recommended that SCOPE be enhanced to provide some type of record-like object. However, because this capability is also similar to N-ary relationships, the theory of N-ary relationships should be investigated before proceeding with the implementation.

Active Attributes, Attribute Triggers

Active attributes and attribute triggers are used by some artificial intelligence tools. These attributes and triggers provide a natural extension to an object-oriented representation. In the logical sense, an active attribute is a derived attribute, or one whose value is derived from the value(s) of another attribute(s). Actually, an active attribute is not an attribute at all, but rather a call to a user-defined function. Whenever its value is requested, the result of executing a user-defined function is returned. An attribute trigger also is attached to a user-defined function, but the function is executed whenever a change is made to the attribute's value. This feature simplifies the construction of user-defined analysis panels and assists users in specifying constraints that keep interdependent attribute values consistent. Attribute triggers also serve as the foundation of a constraint propagation solver. The present SCOPE design permits these features to be added easily.

Better Visible/Invisible Attribute Selection

SCOPE presently can specify the visibility of an attribute to the knowledge base examiner. This capability is useful in reducing display clutter, but it is a coarse setting. A more useful technique is to provide a mechanism where users can define sets of visibility specifications. Users would then be able to select whatever visibility set is appropriate to view just the attributes considered important for the task at hand.

Tiled vs. Overlapping Windows

SCOPE uses tiled views to display each separate tool. While this makes it easier to move between different views, it can waste display space. Therefore, overlapping windows should be investigated as an alternative layout scheme. The provision of a user-configurable frame should also be studied. Such a frame would enable users to set the overlapping windows into a frame so they would operate as the currently tiled SCOPE interface operates.

User Configuration

A number of features that restrict user choice have been designed into SCOPE. It would be helpful if users could select these features and tailor them to their specific needs. Text size, window size and placement, and some color selections could be left to user discretion. Consideration should be given to providing a means for users to define these elements and store them in a user preference file.

Help System

The SCOPE tools could benefit from an on-line help system, such as help commands at the end of each pull-down menu. The help command would explain each command available in that menu. Help for the overall tool would be available from the tool menu.

ACRONYMS

CACC	Concepts Assessment for Combat Control
CINC	Commander-in-Chief
NORAD	North American Air Defense Command
NUSC	Naval Underwater Systems Center
NUWC	Naval Undersea Warfare Center
ODMG	Object Data Management Group
OMG	Object Management Group
OODBMS	Object-Oriented Data Base Management System
PCTE	Portable Common Tools Environment
SCOPE	System Concept of Operations Evaluator
SQL	Structured Query Language
USSPACECOM	United States Space Command

APPENDIX

NORAD AND USSPACECOM COMMAND CENTER STUDY

Background

The North American Air Defense Command (NORAD) and United States Space Command (USSPACECOM) initiated a study to create models of their command centers that would become baselines from which the impacts of crew consolidation, technology insertions, and future missions (e.g., ballistic missile defense) could be studied. At the time the study began, each Command had its own Command Center: the NORAD Command Center (NCC) and the Space Command Center (SPACC). However, the Commander in Chief (CINC) for both Commands (the same person) directed that the NCC and SPACC be consolidated into one facility. This consolidation required that various associated issues be resolved. These issues included development of a joint concept of operations, floor plan/equipment layout, and crew staffing, composition, and tasking. CINC Initiative Funding was provided to develop the command center models from which simulation analysis could be conducted to provide insight into those issues. Existing command center documentation provided a functional and/or process-oriented view of command center responsibilities and operations.

Study Goal

The overall goal of the study was to develop models of the proposed NORAD/USSPACECOM consolidated command center that could be used in the analysis of crew consolidation, impacts of integrating new missions such as Ballistic Missile Defense, and technology insertion (e.g., automated decision aids). The model was to be simulated using standard test and training scenarios to provide quantitative command center component (crew and equipment) workload, utilization, and overall command center throughput performance. Different command center configurations were to be simulated and inter-compared (e.g., levels of consolidation and associated crew member tasking). An additional goal was to make the collected model data (knowledge base) available to decision makers for hands-on inquiry and browsing. These inquiries would in effect be static analyses of the model.

Command Center Overview

This overview provides a general description of the major "pieces" of command centers (i.e., what they did, what functions they accomplished, what physical components made up command centers, and how they responded to specific events [checklists]). In addition, it lists the specific missions of NORAD and USSPACECOM. These "pieces" are the initial inputs into the object decomposition.

As defined in Joint Chiefs of Staff (JCS) Publication (Pub) 1, a command center is "a facility from which a commander and his representatives direct operations and control forces to prosecute their assigned missions. It is organized to gather, process, analyze, display, and disseminate planning and operational data." JCS Pub 19 defines ten generic functions common to most command centers. These functions, which together represent a mission functional decomposition, are: *situation monitoring (8 subfunctions), force and resource status monitoring (5), situation assessment (10)*, operations planning and scheduling (13), *decision making and support (3), mission instruction preparation and dissemination (2)*, employment and execution of forces (7), force employment monitoring (12), and hostility's termination and negotiations (8). The functions in italics are those primarily applicable to the problem domain. The corresponding number of subfunctions is indicated in parentheses. For example, two situation-monitoring subfunctions are: monitor environmental conditions and provide aircraft and missile warning. Each subfunction is associated with specific command center activities and tasks. A complete generic functional decomposition is found in the Command Center Design Handbook (1987).

The NORAD and USSPACECOM command centers were "fixed" command centers operating 24 hours a day, 7 days a week. The command centers consisted of two several-thousand square-foot facilities, mission support equipment, and operational crews with five to eight crew members, each with specific assignments. Mission support equipment is classified as organic (operates within the confines of the command center) or interface (provide information of a specific type from sources external to the command center) (Command Center Design Handbook, 1987). Organic systems include voice communications, information processing, displays, and workstations/consoles. Interface systems include automatic data processing, communications, and warning.

In response to mission-area external events, crew members follow "checklists" that describe step-by-step procedures. Checklists specify what information is needed, its source, decisions to be made, and information to be disseminated. For each checklist item, an action position (crew member) is specified. There are hundreds of checklists that span the operational needs of each command center. Specific NORAD/USSPACECOM checklists exist, for example, for missile launch, SCUD alert, and aircraft hijack.

The missions prosecuted by NORAD were as follows: provide integrated tactical warning and attack assessment (ITW&AA) of missile, air, or space attacks against North America (NA); conduct surveillance and control of NA airspace; provide an appropriate response to an air attack against NA; and detect and perform surveillance on potential narcotic traffickers. The missions prosecuted by USSPACECOM were to do the following: ensure operational control over space systems; provide joint employment of military forces and operational support to other unified combatant commands through space support, force enhancement, space control, and force application for the defense of the U.S. and its space resources; provide space warning and recommended defense measures; and ensure sensor support is provided to NORAD and other CINCs.

Modeling Approach

SCOPE provides direct support for the concept of domain modeling, via the knowledge base layers. Domain modeling provides an effective way to compose a model to obtain a significant amount of reuse of the modeling information. Three domains relating to a command center are the Subject domain, the System domain, and the Concept of Operations domain. The Subject domain is composed of the objects and information related to "what" the system does (e.g., direct air defense). The System domain is composed of the objects and their characteristics that are not related to what the system does (i.e., size, location, and reliability). The Concept of

Operations domain builds on the Subject and System domains by adding attributes relating system components to the activities defined in the Subject domain. Alternative systems would be evaluated by creating alternative system domains and combining them with the already-existing and unique Subject domain to create alternative Concepts of Operations domains. Once defined, each domain would become a knowledge base layer.

This approach to modeling is illustrated in Figure 1. Operational expertise is used to help establish class objects, and their structure and attributes. Device-independent information is decomposed in a subject domain, and implementation-dependent command center information into a system domain. These are combined with man-machine partitioning to form a Concept of Operations domain. When combined with individual tactical scenario data, the Concept of Operations domain becomes a model (static model) of the overall problem domain. Different Concept of Operations domains derived from different task allocation or different system configurations are then inter-compared through static and dynamic analysis. Results of these analyses are fed back into the Concept of Operations domain or System domain (e.g., modifications to configuration, allocation, and communication interfaces).

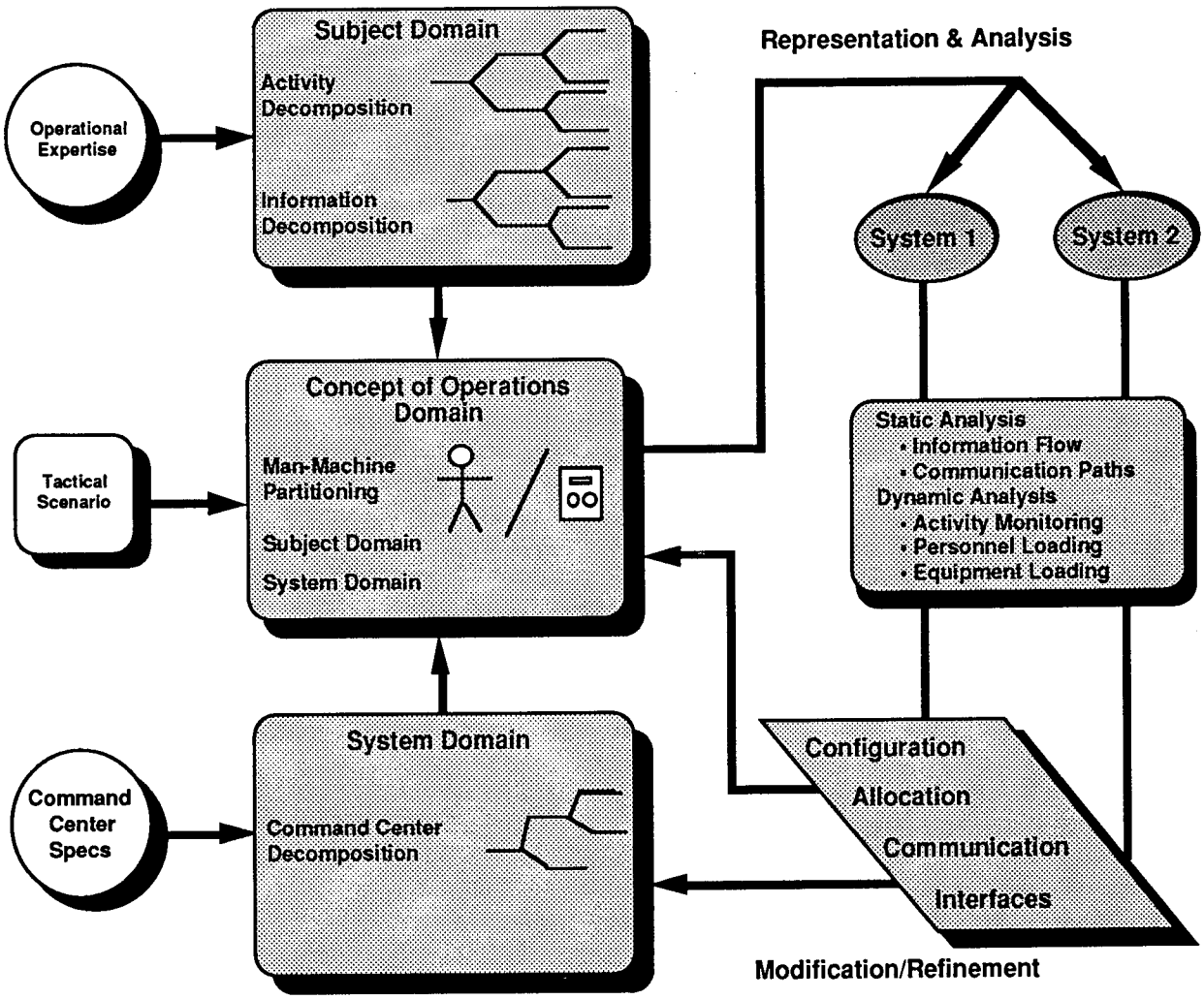


Figure 1
Modeling Approach

Decomposition

The object of the decomposition is to formulate NORAD/USSPACECOM command center descriptions that are compatible with SCOPE. Steps taken to formulate the decomposition include identifying class objects, their substructure, and their attributes. Substructures are the "is-a" and "composed-of" relationships that are relevant to command centers in general and to the problem domain under study specifically. System activities, which are defined as the processes and tasks defined to accomplish a mission function, are treated as a separate class object because SCOPE models activities both as objects (static properties) and as processes (dynamic properties). Table 1 lists the command center class objects and their structures and Table 2 lists the attributes of selected objects. The structure depth (number of levels) of each class object is shown in parentheses. The complete depth or breadth of the structure is not shown. The material shown is meant to be illustrative.

Table 1. Command Center Class Objects and Structures

- Forward User (4 levels deep)
  - National Command Authority
  - U.S. Command
  - U.S. Agency or Organization
- Asset (4)
  - NORAD Region Asset
    - Fighter Force
    - Augmentation Force
    - Surveillance Aircraft
    - Ground-Based Weapon
    - Operation Center
  - USSPACECOM Asset
    - Ground-Based Sensor
    - Space-Based Sensor
    - Space-Based Weapon
- Command Center (2)
  - NORAD Command Center
  - USSPACECOM Command Center
  - Consolidated Command Center
- Command Center Component (4)
  - Crew
  - Mission Support Equipment
    - Communication System
      - Voice
      - Phone
      - Digital
    - Information Processor System
    - Information Display
    - Console/Workstation
- Mission (4)
  - NORAD Mission
  - USSPACECOM Mission
- Event (2)
  - National Threat
  - Theater Threat
  - Conflict
  - Command and Control Security/Integrity Threat
  - Non-Threat
- Information

- Message
- Processed Data
- Command

Activity (3)
- Assess
- Coordinate
- Decide (Decision-Making)
- Employ Force
- Monitor
- Obtain Information
- Plan
- Reconstitute

Staff (2)
- Battle Staff
- Day Staff

Operation Center (2)
- Missile Warning Center
- Space Defense Operation Center

Generic Function (3)

Policy and Regulation

Table 2. Selected Object Attributes

Forward User
- Organization
- Point-of-Contact
- Communication Link
- Location
- Function

Asset
- Assigned Region
- Owner/Operator
- Quantity
- Location
- Operational Status
- Task Status

Command Center
- Floor Plan (Picture)
- DEFCON Level
- Operational Status
- Command Center Component

Command Center Component
- Location
- Tasks

Command Center Component/Crew
- Title (e.g., Command Director)
- Rank
- Nationality
- Security Clearance
- Qualification Level
- Off-Duty Phone
- Duties

Event

- Who, What, When, Where, Why
- Scenario, Script

Information
- Source
- Destination
- Date-Time Stamps
- Capture
- Transmit
- Receive
- Verification
- Log
- Disposition

Activity
- Inputs
- Outputs
- Priority
- Duration
- Termination
- Allocated Resource

References

Beamer, RA.; S.G. Beckner; and S.C. Crawford, 1992. "A Mission-Oriented NUICCS Interface Attributes Analysis," WP920000316, The MITRE Corporation, Colorado Springs, Colorado.

Raymond, T.R. and D.A. Hathaway, 1992. "System Concept of Operations Evaluator (SCOPE)," AL-TR-1992-0161, Air Force Material Command, Wright-Patterson AFB, Ohio.

Shalaer, S. and S.J. Mellor, 1988, "Object-Oriented Systems Analysis: Modeling the World in Data," Yourdon Press, Prentice-Hall.

Signori, D.T., Jr.,1987. "Command Center Design Handbook," DISA, Washington, D.C.